2207/9048

PATENT

UNITED STATES PATENT APPLICATION
FOR

DATA RECOVERY METHOD AND APPARATUS

INVENTORS:

JONATHAN C. LUEKER
DEAN WARREN

PREPARED BY:

KENYON & KENYON
1500 K STREET, N.W.
WASHINGTON, D.C. 20005

(202) 220-4200

Docket No.: 2207/9048

DATA RECOVERY METHOD AND APPARATUS

FIELD OF THE INVENTION

Embodiments of the present invention relate to recovering clock and data information, from serial data.

BACKGROUND OF THE INVENTION

In some data communication arrangements, no separate clock signals are transmitted. This then requires recovering the clock at the receiving end in order to then recover the data. This can be characterized as the problem, in digital communications of, transferring digital signals between multiple clock timing domains. Multiple clock timing domains include the clock timing domain of a transmitting device as well as the clock timing domain of a receiving device. It is not unusual to transmit digital signals between clock timing domains having nearly the same underlying frequency clock, but different or varying phases with respect to each other.

One such arrangement is the Universal Serial Bus (USB). The USB is a bus having electrical, mechanical, and communication characteristics that follows a protocol defined in "Universal Serial Bus Specification" Revision 2.0 published April 27, 2000, by Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation and Koninklijke Phillips Electronics N.V. The USB Specification provides a standardized approach for component interconnection and data transfer.

From the digital communications perspective, a USB transmitting device sends data in the form of packets over a USB cable to a USB receiving device with the clock signal of the transmitting device being used when encoding digital information. Packets include a defined sync field having multiple bits with a transition for each bit (*i.e.*, from a logic 1 to a logic 0 or vis-versa), a payload with data information, and an end of packet field. As

discussed below, the sync field with its transition for each bit, provides a rich set of edges for the USB receiver lock onto the phase and frequency of the USB transmitting device. The USB Specification does not allow for a separate clock signal to be transmitted and this makes it difficult for a USB receiving device to adequately recover the clock signal of the USB transmitting device.

Another problem encountered in digital communications using USB is latching data after an idle period. An idle period is a period of time in which no packets are sent. Once packets are transmitted after an idle state has occurred, the USB transmitter frequency is unknown or may have changed since the previous packets were sent. The USB receiver must be able to recover packets and quickly determine the incoming phase and frequency (*i.e.*, the USB transmitter frequency) without any loss of information. When a packet comes in, the USB receiver has a very short period of time to lock onto the USB transmitter frequency. The shortest time being six bits of sync field, with six bits of sync field only being 12 nanoseconds.

A need, therefore, exists for a technique of transferring signals between multiple clock timing domains that reduces or addresses these problems.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a receiver for receiving digital data signals in accordance with an embodiment of the present invention.

Fig. 2 is a timing diagram for various input and output signals of system components in the embodiment shown in FIG 1.

Fig. 3 is a block/logic diagram of an embodiment of the oversampler of Fig. 1.

Fig. 4 is a block/logic diagram of an embodiment of the sample word register of Fig. 1.

Fig. 5 is a block/logic diagram of an embodiment of the edge detector of Fig. 1.

Fig. 6 is a block/logic diagram of an embodiment of the edge accumulation latches of Fig. 1.

Fig. 7 is a flow diagram of an embodiment of phase selection logic of Fig. 1.

Fig. 8 is a block/logic diagram of an embodiment of the data selector register of Fig. 1.

Fig. 9 is a flow diagram of an embodiment of bit count state machine of Fig. 1.

DETAILED DESCRIPTION

Embodiments of methods and systems for synchronizing data are described. In the following description, for purposes of explanation, numerous specific details are set forth to provide a thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art, that the present invention may be practiced without these specific details. In other instances, structures and devices are shown in block diagram form. Furthermore, one skilled in the art can readily appreciate that the specific sequence in which methods are presented and performed are illustrative and it is contemplated that the sequences can be varied and still remain within the spirit and scope of the present invention.

Fig. 1 is a block diagram of a receiver 10 for receiving digital data signals in accordance with an embodiment of the present invention. As an example, the receiver may comprise a Universal Serial Bus (USB) receiver. However, the method and apparatus of the present invention is applicable to data receivers in which synchronization for data recovery must be performed, in general. As discussed above, in USB signaling, there is no separate clock signal sent over USB cable. There are no unique symbols reserved for start of packet delimiting. Data is sent at a frequency of 480Mb/s with a tolerance of +/- 500 parts per million (ppm).

Between packets, the USB signaling levels on the line go to what is called the idle state. When in the idle state, the lines are both nominally at ground so the differential voltage remains steadily at zero. When signaling in a high speed mode, one of the USB lines is driven nominally 400mV while the other remains at ground. Which of the two lines is high determines whether a 1 or a 0 is being sent. Thus, when in the idle state, the

differential lines voltage is zero, and when driven, the differential line voltage is +/- 400mV.

The receiver of Fig. 1 includes a clock generator 11. The clock generator 11 operates at a clock frequency equal to that of the transmitter from which the data is received within a predetermined tolerance. For example, in the USB this is 480Mb/s with a tolerance of +/- 500 parts per million (ppm), just like the transmitted data. However, the relationship between the phase of the clock and the data being received is unknown. In order to recover the data, this relationship must be determined. Note that when the transmit and receive clock tolerances are combined, the total potential error is the sum of the two, e.g., in USB it can be ± 1000 ppm.

In the illustrated embodiment, the incoming signal on line 13 is coupled into a differential receiver 15. As noted, in a USB system, the nature of the signal is that it is a differential transmission in which one of the lines is high for a logic 1 and the other high for a logic 0. The output of the differential receiver will be a bit stream at the frequency of the transmitter, *e.g.*, 480 MHz. The incoming bit stream is illustrated as timing signal 16 in Fig. 2. As noted above, incoming packets include a sync field having multiple bits with a transition for each bit (*i.e.*, from a logic 1 to a logic 0 or vis-versa) and a payload with data information. What is shown as timing signal 16 represents the sync field where transitions occur for each bit. For data transition is considered to be the case when two adjacent bits do not have the same value.

In the illustrated embodiment, this bit stream is over- sampled by an eight times oversampler 17. Although an eight times over-sampler circuit is shown, any modulus of over-sampler, *e.g.*, 5, can be used without departing from the spirit and scope of the present invention. More generally, what occurs is that data transmitted at a first frequency is over-sampled using an effective clock at a second frequency, n times the receiver clock. In the example, n is eight but could also be some other number. This produces nominally in samples per bit time (also referred to as a unit interval).

As shown in Fig. 2, by the vertical lines 18, in the illustrated embodiment, eight samples extend over a bit time and normally include, at most, a single transition. In other words, a group of the samples will have a logic 1 level and following the transition, a group will have a logic 0 level as indicated by the series of 1s and 0s indicated as 20 on Fig. 2 where boxes are drawn around the groups of eight samples 22a-d. By detecting where that change takes place, it is possible to determine the transition location. Knowing the transition location, a point furthest from the transition can be selected as the point to sample data.

As will be explained in more detail below, in the illustrative embodiment, oversampler 17 provides n outputs, where n=8, *e.g.,* one of the groups 22a-d of eight in Fig. 2, corresponding to the eight samples. These are inputs to a sample word register 19. The sample word register 19 stores m, where, in the illustrated embodiment, m=4, successive groups of eight outputs from the oversampler 17 and outputs them as an (m x n)-bit word at a frequency 1/m that of the data frequency. In the illustrated embodiment, the word is a 32 bit word. (Actually, 40 bits are output, the additional eight bits constituting a duplicate of the oldest eight bit sample). This permits the remainder of the processing to be carried out at this lower frequency greatly reducing the demands placed on the hardware. Again m=4 is only an example and m may be of some other value.

The output from the sample word register to 19, which, in the example shown, comprises 40 samples, is applied to an edge detector 21. Edge detector 21 looks at pairs of these inputs utilizing exclusive or (XOR) gates and thereby determines that which points transitions take place. These outputs of the XOR gates are illustrated as 24a-24c of Fig. 2. The outputs of the edge detector are inputs to edge accumulation latches 23. The outputs of the edge detector will indicate, for five sets of samples at a time, between which of the eight samples a transition took place. Corresponding sample positions in each of the five sets are combined and the results latched. To get a running history, a number of latches are utilized. The edge accumulation latches 23 also receive an input from a packet state

machine 20 which in turn receives an input from a squelch detector 27 having as its input the incoming signal line 13.

The purpose of the squelch detector, in this particular embodiment, is to determine the beginning of the transmission of a packet of data. It senses whether or not there is a signal on either of the differential lines. If no data is being transmitted, both lines will be at ground. The packet state machine is utilized to provide the necessary control signals, for example, to the edge accumulation latches 23, to insure they are reset as necessary to maintain an accurate history.

Ins A3

The output of the edge accumulation latches is provided as input to phase selection logic 29. In a manner to be described below, in the illustrative embodiment, the phase selection logic 29 selects, as the desired phase, the phase that is the greatest distance from the transitions. In other words, it is desired to select a phase for sampling data that is as close to the middle of the bit time as possible. The determined phase is output to the data selection register which uses it to select the data sample at that phase for each bit. The data selection register obtains the data from the sample word register 19.

The phase selection logic 29 also provides an output to a bit count state machine 33. Normally, in the illustrative embodiment, the data selection register outputs four valid bits. However, because of the tolerance on the frequency of the transmitter clock and receiver clock, it is possible that the phase will drift. This may take place in either direction. As a result, in some cases the data selection register will output only three valid bits and in other cases it may output five valid bits. The bit count state machine provides an indication of the number of valid bits being output by the data selection register 31.

Ins A4

Fig. 3 is a diagram of an embodiment of an over-sampler 17 which may be used in embodiments of the present invention. Serial data on line 51 is input to the data input of, for example, D-type flip flops 53a through 53h. Each of these is clocked by a clock signal from the clock generator 11 of Fig. 1 at the bit frequency. However, each is at a different

phase. Thus, for example, flip flop 53a is clocked at zero phase, 53b at a phase of 45 degrees, 53c at a phase of 90 degrees, and so on, each phase being shifted 45 degrees from the adjacent phase. These clock phases are shown as 54a (0 deg.), 54b (45 deg.) and 54h (315 deg.) on Fig. 2. In-between clock 54b and clock 54h are additional clock for the intervening phases. The lines 18 of Fig. 2 correspond to the leading edges of these eight clock signals.

After going through the eight phases, with eight samples at the outputs of the eight flip flops 53a through 53h, these outputs are latched in another set of eight flip flops 55a through 55h by the zero phase clock. The resulting output comprises eight data signals representative of the logic level of the data at each of the eight sampling phases as shown, for example by group 22a of Fig. 2.

INSERT A5 Fig. 4 is a diagram of an exemplary embodiment of sample word register 19 of Fig. 1. The input to this unit comprises the eight lines output by the over-sampler 17. These eight lines are coupled as inputs to each of four byte registers 61a-61d. The clock generator 11 of Fig. , in addition to generating the eight phases at the data rate, for example, 480MHz, also generates 4 clocks at 120MHz. These are shown as 62a-62d in Fig. 2. Although at the same frequency, these clocks are delayed by different amounts. The first clock 62a has no delay, the second 62b, a delay of 90 degrees, the third 62c, a delay of 180 degrees and the fourth 62d, a delay of 270 degrees. As can be seen from Fig. 2, this results in clock edges which effectively, when taken as a group, occur at a 480MHz rate. Thus, after the data has been latched into the eight latches 55a-55h of Fig. 3, which data appears on line 63, a 120MHz clock 62a transition will clock that data into byte register 61a. Then, eight more samples of data are obtained for the next data bit. This signal on line 63 is then clocked into byte register 61b by the 120MHz clock 62b delayed by 90 degrees. Similarly, the next two sets of samples are clocked into byte registers 61c and 61d by clock 62c and 62d. The output of byte register 61d is provided as an input to a byte register 61e. This byte register is also clocked by the 120MHz clock at zero phase. As a result, when the data is clocked into byte register 61a, at the same time, the data which

had previously been stored into byte register 61d is clocked into byte register 61e. Subsequently, the data at the output of byte registers 61a-61e is clocked into a set of byte registers 65a-65e.

The result at the output of byte registers 65a-65e, which is the output of the sample word register 19, in the present example, is a set of 40 samples. 32 of the samples represent eight samples for each of four successive data bits, with the remaining eight being the last eight bits from the previous 32 samples. In other words, all of the samples 22a-22d, along with a previous eight samples 22e will be output at the same time. As noted above, this output is provided to the edge detector.

An embodiment of an edge detector 21 which can be used in the embodiment of Fig 1 is illustrated in Fig. 5. The edge detector comprises a plurality of 40 XOR gates 71. Only the gates for edges 0-7 and 39 are illustrated. It will be understood that between the gate for edge 7 and edge 39, gates for edges 8-38 will be provided. XOR gate 71a has as its inputs data 0 and data 31 from the previous sample word. Similarly, gate 71b has as inputs data 0 and data 1. In similar fashion, adjacent data lines provide the inputs to the remaining XOR gates. Since an XOR gate provides a logic one output only in the case where only one of its inputs is a logic one, an output will occur only where there is a transition between the two data inputs. That is, an output will occur only when one of the inputs is at a logic zero level and the other at a logic one level indicating a change.

Fig. 6 shows an exemplary embodiment of the edge accumulation registers of Fig. 1. The edge outputs of the embodiment of the edge detector of Fig. 5 provide inputs to the edge accumulation register of Fig. 6. The 40 edge outputs comprise five groups of eight outputs. Three of these groups are shown as 24a-24c in Fig. 2. The relative positions of edge 0 in the first group 24a is the same as that of edge 8 in the second group 24b, edge 16 in the third group 24c, edge 24 in the fourth group and edge 32 in the fifth group. Thus, in the edge accumulation register unit 8, OR gates 81a-81h are provided, one for each of the eight possible edge positions within a bit time. In Fig. 6, only gate 81a, corresponding to

the first edge position, and gate 81h, corresponding to the last edge position are illustrated. In between will be six more such gates.

A particular OR gate will have an output if any of its inputs is at a logic one level. Thus, if in any of the five groups of edges, the first edge position, *i.e.*, 0, 8, 16, 24 or 32 is at logic 1, the gate will have an output indicating a transition at the first edge position. In the example illustrated, where the difference in frequency between data 16 and the clocks 54a-54h has been exaggerated, position 3 in the first group 24a has a transition, position 12 in the second group 24b has a transition and position 21 in the third group 24c has a transition. Thus, gates 81d, e and f will all have outputs at a logic one. Some of the remaining gates may also have outputs due to the inputs not shown.

The outputs of the 8 gates 81a-81h are inputs to each of four byte latches 85a-85d. These are essentially set-reset latches with the outputs of the OR gates 81a-81h setting the latches. Although indicated as a single latch, each of latches 85a-85d actually comprises 8 latches, one coupled to each of the OR gate 81a.-81h outputs. Thus, in the example given, at least the three latches at positions 3, 4 and 5 would be set.

When the beginning of a packet is detected, all of the byte latches 85a-85d are reset and begin accumulating data. Initially, all of the byte latches 85a-85d will contain the same information. That is, the position connected with a detected edge transition(s), *e.g.*, 3, 4 and 5 will be set in each. More than one may be set, for example, due to jitter. In any case, as time passes, if, for example, the clock frequencies differ, the edge location will drift and be at a different location. The exaggerated offset in clock frequency introduced into Fig. 2 has the effect of showing what happens over time with drift. Then multiple outputs from the byte register will be at a logical 1 level as was seen above. Eventually, all edges will be high and it becomes impossible to select the location which is furthest away from the transitions.

In SA67

Thus, the latches are reset on rolling basis, under control of the packet state machine 25. For example if latch 85a is reset, it will then begin to accumulate only the most recent information. If, in succession, latch 85b, then 85c and then 85d are reset, latch 85e will have the oldest information 85a next oldest and so on. Thus, a rolling history of edge transitions is maintained. The outputs of the latches, of which there will be eight for each of latches 85a-85e, are provided into a set of 8 OR gates 87a-87h. These OR gates then provide a historical indication of edge transitions.

Fig.7 is a flow diagram of an embodiment of phase selection logic of Fig. 1. The outputs of the OR gates 87a-87h are combined such that each of the history bits is Ored with the following bit as indicated in block 91. As also indicated, the latest is Ored with the earliest. This will result in 8 outputs. These 8 outputs form the highest byte on line 93. The 8 bit outputs on line 93 are again Ored, in a further 8 Or operations. As indicated in block 95, each of the 8 outputs on line 93 is Ored with the bit on either side. The first and last bits wrap around. The output on line 97 from the Oring function in block 95 is the second highest byte. The 8 outputs from block 95 are provided to another block 99 where another 8 OR operations take place. This is carried out in the same manner as in block 95. The resulting output on line 101 line is the third highest byte.

The highest byte, second highest byte and third highest byte are inputs to logic 103. As indicated, this logic determines the highest level byte which has two or fewer zeros. As indicated, it outputs the phase of the sole remaining 0 if there is only one 0 remaining. It outputs the phase of the latest 0 if only an adjacent pair remains. It outputs an error indication if 2 non-adjacent zeros remain. The output from logic 103 is an input to block 105. Block 105 limits the maximum phase change to plus or minus one sample phase per iteration. The output of block 105 on line 107 will be the selected sample phase. That is to say, it will be one of the phases 54a-54h of Fig. 2.

Fig. 8 is a block/logic diagram of an embodiment of the data selector register of Fig. 1. This embodiment of the data selector register comprises four 8-input multiplexers

111a to 111d, of which only the first and last are shown. These receive inputs from the byte registers 65a-65e of Fig. 4. In addition, they receive a select input from line 107 of Fig. 7. This will be a three bit signal permitting selection of one of the 8 sample inputs to each of the multiplexers 111a-111d.

Fig.9 is a flow diagram of an embodiment of the bit count state machine of Fig. 1. In block 120, a comparison is made to see if the current phase selection is different from the previous phase selection. If it is not, then the four bits 4:1 are valid and an appropriate indication is provided on line 121. If the current phase selection is different, block 122 is entered. Here, a check is made to see whether the current phase selection "wraps around" from the previous phase. If it does not, then four bits 4:1 are valid and an appropriate indication is provided on line 124.

If the phase did wrap around, then the answer from block 122 is "yes" and block 126 is entered. Here a check is made to see if the current phase selection "wraps around" from the latest to the earliest sample phase. If so, then the three bits 3:1 are valid and this is indicated on line 128. If not, block 130 is entered at a check made to see if the current phase selection "wraps around" from the earliest to latest sample phase. If it does, an output on line 132 indicates that the five bits 4:0 are valid.

Thus, the final outputs of the data selector register in bit count state machine of Fig. 1, provides 5 bits of output data and an indication of which of those bits are valid. This information can then be subsequently used by other circuits.

Embodiments of methods and apparatus for clock and data recovery have been described. In the foregoing description, for purposes of explanation, numerous specific details are set forth to provide a thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the present invention may be practiced without these specific details. In other instances, structures and devices are shown in block diagram form. Furthermore, one skilled in the art can readily appreciate that the specific

sequences in which methods are presented and performed are illustrative and it is contemplated that the sequences can be varied and still remain within the spirit and scope of the present invention.

In the foregoing detailed description, apparatus and methods in accordance with embodiments of the present invention have been described with reference to specific exemplary embodiments. Accordingly, the present specification and figures are to be regarded as illustrative rather than restrictive.